IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Applicants:	Kosche et al.	
App. No.:	10/840,164	Con. No.: 7512
Filed:	May 6, 2004	Art Unit: 2192
Title:	METHOD AND APPARATUS FOR PROFILING DATA ADDRESSES	Examiner: Tecklu, Isaac Tuku

AMENDMENT AND RESPONSE TO FINAL OFFICE ACTION

MAIL STOP AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In response to the final Office action dated October 16, 2008, please consider the following remarks and amend the above-identified application as follows:

Amendments to the Claims begin on page 2 of this paper.

Remarks begin on page 9 of this paper.

Amendments to the Claims:

1. (Currently Amended) A tangible computer readable storage medium comprising a computer-implemented software tool that determines at least one data address from one or more instruction instances, and that identifies one or more memory reference objects, which are associated with the data address, as hindering execution of code that includes the instruction instances, wherein the instruction[[s]] instances correspond to the code execution hindrance, wherein the memory reference objects include virtually addressable memory.

2. (Original) The software tool of claim 1 wherein the memory reference objects include one or more of physical memory reference objects and logical memory reference objects.

3. (Original) The software tool of claim 2 wherein the memory reference objects include one or more of a cache, cache lines, cache levels, cache sub-blocks, memory controllers, addressable memory, and memory-management page translation units.

4. (Cancelled)

5. (Original) The software tool of claim 2 wherein the logical memory reference objects include one or more of source-level data objects, memory segments, heap variables, variable instances, and stack variables.

6. (Original) The software tool of claim 5 wherein the source-level data objects include one or more of functions, statically linked objects, data structures, data types, data type definitions, operands, and expressions.

7. (Original) The software tool of claim 6 wherein the statically linked objects include one or more of global variables and static variables.

8. (Original) The software tool of claim 1 wherein the software tool includes one or more of a compiler, an interpreter, an optimization tool, and a virtual machine.

9. (Original) The software tool of claim 1 wherein the code includes one or more of machine code, byte code, and interpreted code.

10. (Original) The software tool of claim 1 that also aggregates addresses based on the memory reference objects.

11. (Original) The software tool of claim 10 wherein the software tool utilizes at least a portion of the data addresses to aggregate the addresses.

12. (Original) The software tool of claim 10 that also provides the aggregated addresses and an indication of the code execution hindrance corresponding to the aggregated addresses for one or more of storage and display.

13. (Cancelled)

14. (Original) The software tool of claim 1 wherein the code execution hindrance corresponds to one or more sampled runtime events.

15. (Original) The software tool of claim 14 wherein the sampled runtime events include one or more of cache misses, cache references, data translation buffer misses, data translation buffer references, and counter condition events.

16. (Currently Amended) A method for profiling code executing in a computer system, the method comprising:

identifying an instruction instance that corresponds to a runtime event;

determining a data address from the instruction instance; and

determining a memory reference object from the determined address;

wherein the data address includes a virtual address in the computer system.

17. (Original) The method of claim 16 wherein the runtime event is a sampled runtime event.

18. (Original) The method of claim 16 wherein identifying the instruction instance comprises backtracking from a second instruction instance to the instruction instance.

19. (Original) The method of claim 16 wherein determining the address from the instruction instance comprises decoding the instruction instance.

20. (Original) The method of claim 19 further comprising:

decoding the instruction instance if a register that hosts the instruction instance is

determined as valid.

21. (Original) The method of claim 20 wherein determining if the register is valid comprises:

applying reverse register transformation with respect to the runtime event; and

determining whether the register is valid based on the applied reverse register transformation.

22. (Original) The method of claim 16 wherein the memory reference object includes a physical memory reference object or a logical memory reference object.

23. (Original) The method of claim 22 wherein the physical memory reference object includes cache, a cache line, a cache sub-block, a cache level, a memory controller, or a memory-management page translation unit.

24. (Original) The method of claim 16 wherein the logical memory reference object includes a source-level data object, a memory segment, a heap variable, or a stack variable.

25. (Original) The method of claim 24 wherein the source-level data object includes a data type, a data type definition, a statically linked object, an operand, a data structure, or an expression.

26. (Original) The method of claim 25 wherein the statically linked object includes a global variable or a static variable.

27. (Original) The method of claim 16 wherein the instruction instances include memory accessing instructions.

28. (Cancelled)

29. (Original) The method of claim 16 further comprising aggregating a plurality of addresses that include the determined address, based on the memory reference object.

30. (Original) The method of claim 29 wherein the aggregating of the plurality of addresses utilizes at least a portion of the addresses.

31. (Original) The method of claim 29 further comprising providing the aggregated plurality of addresses for one or more of display, storage, and manipulation.

32. (Previously Presented) The method of claim 16 embodied as a computer program product encoded on one or more machine-readable physical storage media.

33. (Currently Amended) A method of profiling code executing in a computer system, the method comprising:

associating data addresses with memory reference objects, wherein the data addresses have been determined from instruction instances corresponding to code execution hindrance; and

aggregating the data addresses based on their associated memory reference objects;

wherein the data addresses include virtual addresses in the computer system.

34. (Original) The method of claim 33 wherein the instruction instances include memory accessing instructions.

35. (Original) The method of claim 33 wherein the code execution hindrance corresponds to one or more runtime events.

36. (Original) The method of claim 35 wherein the runtime events are sampled runtime events.

37. (Original) The method of claim 35 wherein the runtime events include one or more of counter condition events, cache misses, cache references, data translation buffer references, and data translation buffer misses.

38. (Cancelled)

39. (Original) The method of claim 33 wherein said aggregating utilizes at least a portion of the data addresses.

40. (Previously Presented) The method of claim 33 embodied as a computer program product encoded on one or more machine-readable physical storage media.

41. (Currently Amended) A method of profiling code in a computer system

comprising:

identifying an instruction instance corresponding to a runtime event;

determining whether the instruction instance is valid;

decoding the instruction instance to extract at least a portion of a data address if the instruction instance is valid;

determining a memory reference object with the extracted portion of the address; and

aggregating the data address with other addresses based at least in part on the memory reference object, wherein the memory reference object includes virtual addresses in the computer system.

42. (Original) The method of claim 41 further comprising associating the extracted portion of the data address with the memory reference object.

43. (Cancelled)

44. (Original) The method of claim 41 wherein the runtime event is a sampled runtime event.

45. (Original) The method of claim 41 further comprising:
applying reverse register transformation with respect to the runtime event to determine if the instruction instance is valid.

46. (Previously Presented) The method of claim 41 embodied as a computer program product encoded on one or more machine-readable physical storage media.

47. (Currently Amended) A computer program product for profiling code, encoded on one or more machine-readable physical storage media, the computer program product, which when executed, performs operations comprising:

identifying a valid instruction instance that corresponds to a runtime event;

determining a data address from the identified valid instruction instance;

determining a memory reference object with the determined data address; and

aggregating a set of addresses, which include the determined data address, based at least in part on the memory reference object, wherein the memory reference objects include virtually addressable memory.

48. (Original) The computer program product of claim 47 wherein the operations further comprise associating the determined data address with the memory reference object.

49. (Original) The computer program product of claim 47 wherein the operations further comprise:

applying reverse register transformation with respect to the runtime event; and determining if the instruction instance is valid from the applied reverse register transformation.

50. (Original) The computer program product of claim 47 wherein the memory reference object includes a physical memory reference object or a logical memory reference object.

51. (Cancelled)

52. (Original) The computer program product of claim 50 wherein the logical memory reference object includes a source-level data object, a memory segment, a heap variable, a variable instance, and a stack variable.

53. (Original) The computer program product of claim 52 wherein the source-level data object includes a data type, a data type definition, an operand, a statically linked object, a data structure, or an expression.

54. (Original) The computer program product of claim 53 wherein the statically linked object includes a global variable or a static variable.

55. (Currently Amended) An apparatus comprising:

a processor;

memory; and

means for identifying a memory reference object and identifying a data address corresponding thereto from an instruction instance that corresponds to one or more runtime events, and aggregating a set of addresses that include the data address, based at least in part on the memory reference object, wherein the memory reference object includes virtually addressable memory.

56. (Original) The apparatus of claim 55 wherein the memory reference object includes a physical memory reference object or a logical memory reference object.

57. (Cancelled)

58. (Original) The apparatus of claim 56 wherein the processor includes event condition counters.

REMARKS

The Applicants have received the Office action dated October 16, 2008, which 1) rejects claims 1-15 under 35 U.S.C. § 101; 2) rejects claims 16-58 under 35 U.S.C. § 102 over Yates et al. (U.S. Patent No. 7,111,290) hereinafter "Yates".

With this Response, the Applicants amend claims 1, 16, 33, 41, 47, and 55, and cancel claims 4, 13, 28, 38, 43, 51, and 57.[1] Therefore, after entry of this Response, claims 1-3, 5-12, 14-27, 29-37, 39-42, 44-50, 52-56, and 58 remain pending.

I. Claim Rejections Under 35 U.S.C. § 101

The Office action rejects claims 1-15 as allegedly directed to non-statutory subject matter. Without conceding the merits of the rejection and solely to advance prosecution, independent claim 1 is amended to recite a "tangible computer readable storage medium comprising a computer-implemented software tool.[2]

The Applicants respectfully submit that the claims, as amended, are directed to statutory subject matter, in compliance with 35 U.S.C. § 101 and respectfully request such indication.

II. Claim Rejections Under 35 U.S.C. § 102

The Office action rejects claims 1-58 under 35 U.S.C. § 102(e) as allegedly anticipated by Yates. "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). The Applicants respectfully traverse the § 102 rejections because Yates simply does not teach or suggest "each and every element as set forth in the claim" either expressly or inherently.

For example, independent claim 1, as amended, requires (emphasis added)

> a computer-implemented software tool that determines at least one data address from one or more instruction instances, and that identifies one or more memory reference objects, which are associated with the data address, as hindering execution of code that includes the instruction instances correspond to the code execution hindrance, wherein the memory reference objects include virtually addressable memory.

Yates appears to be directed to a method of profiling program execution to identify frequently and infrequently executed regions of the program to be determined (*see Yates, column 2, lines 14-25*), and not necessarily identifying hindering code. Nevertheless, even if Yates did teach identification of hindering code, Yates fails to teach or suggest that the

[1] Support for these amendments is found at least in original claim 3.
[2] *Ex parte Bo Li*, Appeal 2008-1213 (BPAI 2008) (affirming *Beauregard* claims post *In re Bilski*).

location within the computer system at which program code is being hindered is a virtual memory location as required by claim 1. In fact, Yates teaches the opposite: "profiler 400 tracks events by physical address, rather than by virtual address". *Yates, col. 55, ll. 47-48 (emphasis added); see also col. 29, 7-9.* For at least this reason, claim 1 and its dependent claims (claims 2-15) are not anticipated by Yates.[3]

Independent claims 16, 33, 41, 47, and 55, as amended, contain limitations akin to claim 1, and therefore, claims 16, 33, 41, 47, and 55, and their dependent claims (claims 17-27, 29-32, 34-37, 39, 40, 42, 44-46, 48-50, 52-54, 56, and 58) are not anticipated by Yates for at least the same reason as claim 1.

III. Conclusion

The Applicants thank the Examiner for his thorough review of the application. The Applicants respectfully submit the present application, as amended, is in condition for allowance and respectfully request the issuance of a Notice of Allowability as soon as practicable.

The Applicants believe no fees or petitions are due with this filing. However, should any such fees or petitions be required, please consider this a request thereof and authorization to charge Deposit Account No. 04-1415 as necessary.

If the Examiner should require any additional information or amendment, please contact the undersigned attorney.

Dated: Dec. 16, 2008

Respectfully submitted,

Gregory P. Durbin, Registration No. 42,503
Attorney for Applicant
USPTO Customer No. 66083

DORSEY & WHITNEY LLP
Republic Plaza Building, Suite 4700
370 Seventeenth Street
Denver, Colorado 80202-5647
Phone: (303) 629-3400
Fax: (303) 629-3450

[3] Furthermore, since Yates teaches directly against a requirement of claim 1, it is difficult to imagine how Yates could be combined with another reference, for example in a future obviousness rejection, to render the claims obvious.